A real-time hybrid neuron network for highly parallel cognitive systems

Christiaanse, G.J.; Zjajo, A.; Galuzzi, C.; Leuken, R. van

DOI
10.1109/embc.2016.7590820

Publication date
2016

Document Version
Accepted author manuscript

Published in
2016 IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society (EMBC)

Citation (APA)
Christiaanse, G. J., Zjajo, A., Galuzzi, C., & Leuken, R. V. (2016). A real-time hybrid neuron network for highly parallel cognitive systems. In *2016 IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society (EMBC)* (pp. 792-795). IEEE. https://doi.org/10.1109/embc.2016.7590820

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

A Real-Time Hybrid Neuron Network for Highly Parallel Cognitive Systems

Gerrit Jan Christiaanse*, Amir Zjajo*, Carlo Galuzzi†, Rene van Leuken*
*Circuits and Systems Group, Delft University of Technology, Delft, The Netherlands
†BMI Group, Maastricht University, Maastricht, The Netherlands
{g.j.christiaanse, a.zjajo, t.g.r.m.vanleuken}@tudelft.nl
c.galuzzi@maastrichtuniversity.nl

***Abstract*—For comprehensive understanding of how neurons communicate with each other, new tools need to be developed that can accurately mimic the behaviour of such neurons and neuron networks under 'real-time' constraints. In this paper, we propose an easily customisable, highly pipelined, neuron network design, which executes optimally scheduled floating-point operations for maximal amount of biophysically plausible neurons per FPGA family type. To reduce the required amount of resources without adverse effect on the calculation latency, a single exponent instance is used for multiple neuron calculation operations. Experimental results indicate that the proposed network design allows the simulation of up to 1188 neurons on Virtex7 (XC7VX550T) device in brain real-time yielding a speed-up of x12.4 compared to the state-of-the art.**

I. Introduction

Various parts of the human brain have particular areas of focus. The *Neuron Network* of the *Inferior Olive* discussed in this paper deals with the brain responses to reflexes, or, more directly, how the brain processes signals that reflect complex movements and motor-coordination [1]. Neuron networks consist of thousands/millions of neurons, which are highly interconnected via synapses used to transmit signals to individual target cells. Several neuron network models have been proposed [2]–[6], which replicate cell and network behaviour in various degrees of complexity, accuracy [7] and characteristics, e.g. spike-train amplitude, frequency, precise arrival times. Multi-core software designs [8], [9] have proved to be capable of simulating large neuron networks within a given time. Due to the high level of parallelism in neuron networks, reconfigurable hardware, such as a *Field Programmable Gate Arrays* (FPGA), however, provide the means for real-time and even hyper-real-time simulation of these intricate and highly parallel networks [10]. Additionally, the reconfiguration property of FPGAs provides the flexibility to emulate the plasticity of neuron networks and to modify the brain models on demand. The behaviour of a highly parallel cognitive system, such as the *Inferior Olive*, can be accurately simulated with a mathematical model that closely resembles the biological responses in the human brain [11]. The extended *Hodgkin-Huxley model* describes the relation between the electric current to a single neuron membrane, and its capacitance. This relation is translated into nonlinear differential gap functions [12] that describe the responses of three main parts of a neuron: the dendrite, the soma and the axon hillock. These functions rely a great deal on accurate floating point operations[1]and, in particular, on the exponent operation. Within the Hodgkin-Huxley model equations, the exponent operation needs to be executed 30 times per neuron calculation. Compared to standard operations, the exponent operations require relatively more resources and cycles to complete.

In this paper, in order to efficiently solve these problems, we propose a new approach based on the use of a single exponent instance over multiple neuron calculations in a Kahn process network [13]. Hence, the required amount of resources is reduced without having an effect on the calculation latency. Furthermore, if the computational requirements were to increase, more exponent instances can be added to meet this requirement. The contributions of the work presented in this paper can be summarised as follows:

- the design and implementation of a scalable, real-time[2] and biophysically meaningful neuron network using both single and double floating-point precision;
- the application of an open source floating-point IP block, increasing the portability of the design to multiple FPGA targets.

II. The Hybrid Neuron Network for Highly Parallel Cognitive Systems

Through the Hodgkin-Huxley model, each Inferior Olive neuron in the neuron network receives a coupling effect from its neighbouring neurons (Fig. 1). From the coupling effect, a possible impulse, and the current Inferior Olive neuron states, new state voltages are generated [14]. We refer to each (neuron) node in the neuron network as a *Simulated Cell (SimC)*, whereas we refer to the hardware used to simulate the cells as a *Physical Cell (PhyC)*. The idea is to partition the cells in optimised clusters, called *Physical Cell Clusters*, where a number of cells shares a certain amount of resources. In the physical cell clusters, configurable routing tables are responsible for how the simulated cells are arranged within the neuron network. By attaching each physical cell cluster

The work presented in this paper is supported by the European Union and the Dutch government, as part of the CATRENE program under the Heterogeneous INCEPTION project.

[1]In [7], it is shown that using 32-bit floating-point precision instead of 64-bit floating-point precision does introduce a small error, although it does not seem to affect the overall behaviour of the neuron and its response.

[2]The 'real-time' constraint, with a step time of 50 µs is given by [14].

978-1-4577-0220-4/16/$31.00 ©2016 IEEE

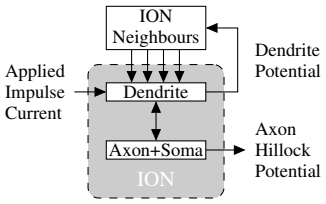

Fig. 1: Simple Inferior Olive neuron architecture.

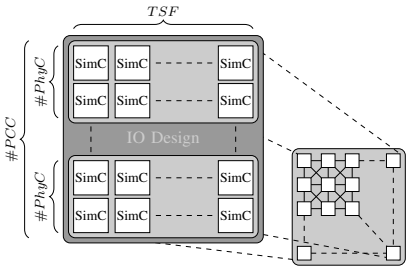

Fig. 2: A configured Inferior Olive design, mapped to a desired network topology.

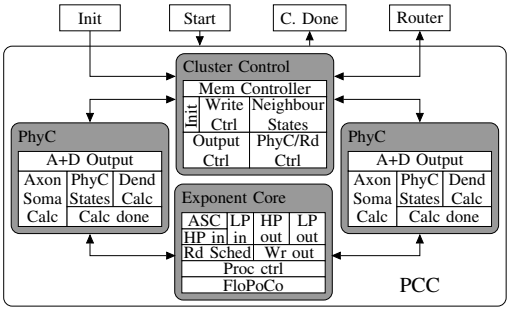

Fig. 3: Physical cell Cluster configured with 2 physical cell's and 1 Exponent Coprocessor

to a binary tree network, responses between simulated cells are shared. Furthermore, through the top node of the tree network, a current impulse can be applied to the simulated cells, and all output results of the neuron network are streamed. The design can be tuned using 4 parameters: the number of physical cell clusters, the number of physical cells in each cluster, the amount of shared *Exponent Coprocessors* within each physical cell cluster, and the *Time Sharing Factor* (TSF) for each physical cell. If a physical cell can calculate multiple responses within a given *Time Step* (T_{Step}), the TSF will be larger than one. Fig. 2 illustrates a configured Inferior Olive design, mapped to the desired neuron network topology[3].

A. Optimised design

In the Inferior Olive design, one or several physical cell cluster's are interconnected by a tree network and implemented on an FPGA. In Fig. 3 the main parts of the physical cell cluster are shown. A physical cell cluster consists of three main elements: *Cluster Control*, physical cell, and the exponent coprocessor. The cluster control can initialise the physical cells, stores the neighbouring neuron states, controls the communication inside and outside the physical cell cluster, and distributes the needed data to the physical cells through several smaller controllers. The physical cells compute the next axon hillock and dendrite potentials from the current simulated cell states, and neighbouring cells dendrite potential. An important element is a coprocessor, based on an open source core (FloPoCo) [15], that computes the exponent calculations for its connected physical cells. As all calculations are pipelined within the coprocessor, any Inferior Olive network topology independent scheduled calculations are grouped in order, and also pipelined through an *Application Specific Coprocessor* to save resources within the physical cell. Communication to and from the coprocessor are handled by high and low priority in- and output FIFO buffers. Via a read scheduler, the input signals are given to the FloPoCo, after which a write controller retrieves the calculated exponent and writes it back to the output FIFO buffer, respectively.

B. Schedule

After a *Start* signal is sent to the neuron network, each physical cell will calculate the next states for a certain set of simulated cells, before sending a *Calc Done* signal back. Within the physical cell, the topology dependent *Dendrite Calculation* and topology independent *Axon+Soma Calculation* run in parallel (Fig. 4). Internally, the dendrite calculation is dependent on the result of the axon+soma calculation to calculate the new dendrite state. Externally, both calculations use the same exponent coprocessor. As the axon+soma calculation has a longer critical path and is not dependent on the topology, it is scheduled with a higher priority over the dendrite calculation by the read scheduler in the exponent coprocessor. Furthermore, each axon+soma calculation within a physical cell cluster is synchronised, resulting in axon hillock potentials being calculated at predictable times. The dendrite calculation is however, not synchronised, giving it more flexibility over the exponent coprocessor. By keeping the critical path within the dendrite calculation to a minimum, and by allowing it to start processing new network neighbours before the current exponent is known, each simulated cell can quickly be scaled up to allow more connections within the neuron network. The exponent coprocessor is thus constantly being given new values to calculate, with high priority tasks arriving after a deterministic amount of cycles.

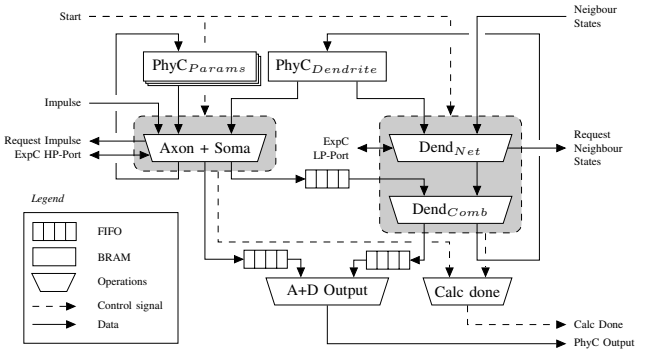

Fig. 4: Data flow of a physical cell. The box on the left is the Axon/Soma Calc. The one on the right is the Dend Calc. Cell states are stored in BRAM. All in- and output data signals are connected to FIFO buffers (not shown).

C. Communication

When a physical cell has calculated a new axon or dendrite potential for a given simulated cell, it is sent through the tree network by the cluster control. The axon potentials are only directed to the I/O of the FPGA, while the dendrite values are sent to all of the other physical cell clusters. If, during this time a simulated cell needs to be released from/receive an impulse, or afterwards a dendrite value needs to be overwritten, a signal can be injected into the tree

[3]In the current configuration, each simulated cell neighbours 8 other simulated cells.

network. When all responses have been streamed to the I/O of the FPGA and received by the clusters, respectively, the design is ready to start calculating the new simulated cell states.

III. Design configuration

Our main goal is to optimise and maximise the use of hardware resources in the FPGA to be able to increase the size of the simulated neuron network. To find an optimal design, first a limit is given to the total amount of implementable physical cells ($\#Tot_{PhyC}$) in the FPGA, based on the critical resources. By dividing this value by the *grouping factor* (φ), the amount of physical cell clusters ($\#PCC$) is determined

$$\#Tot_{PhyC} < \frac{\#_{Crit\ Resources}}{\#_{Resources,PhyC}} \tag{1}$$

$$\#PCC = \left\lfloor \#Tot_{PhyC} \times \frac{1}{\varphi} \right\rfloor \tag{2}$$

To simulate the biological behaviour of a cell, each physical cell requires a certain number of cycles (C_{phyc}), which is determined by the topology dependent (C_{dend}) and independent (C_{a+s}) computation time (in cycles), respectively,

$$C_{phyc} = max(C_{dend}, C_{a+s}). \tag{3}$$

Each neuron response is governed by combination of the results of the topology dependent and independent calculations. By describing the latency of the topology dependent calculation, the C_{phyc} can be written as function of C_{dend}[4]

$$C_{dend} = max(C_{din}, C_{a+s}) + \tau. \tag{4}$$

The topology dependent calculation, as a function of the latency of the dendrite calculation to process network inputs (C_{din}), can be split in three sections

$$\begin{aligned} C_{din} &= \delta(\varphi) + \ max(C_{block}, \alpha \times N_D) + \omega \times N_D, \\ N_D &= \left\lceil \frac{N}{\#Dend} \right\rceil, \quad \omega \geq \left\lceil \frac{\#PhyC \times \#Dend}{\#ExpC} \right\rceil. \end{aligned} \tag{5}$$

$$\begin{aligned} C_{block} &\leq \left\lceil 22 \times \frac{\#PhyC}{\#ExpC} \right\rceil + \left\lfloor \frac{\#Dend_{Clus} - 1)}{\#ExpC} \right\rfloor + \rho, \\ \#Dend_{Clus} &= \#PhyC \times \#Dend. \end{aligned} \tag{6}$$

Firstly, we calculate the start-up delay δ, which is partly dependent on the amount of dendrite calculations that share a memory core. Next, we find the amount of cycles α that take place before each (low priority) exponent calculation and, finally we calculate the number of cycles ω after the result of the exponent calculation is known. The calculation of the exponent itself is being carried out by the exponent coprocessor with pipeline depth ρ. If the exponent calculation is being blocked by another task after all α calculations are performed, an extra blocking time has to be taken into account (6). $\#Dend$ are the amount of Dend_{Net} operations housed in each physical cell[5]. Increasing the number of exponent coprocessors per physical cell cluster ($\#ExpC$) (together with the $\#Dend$) with a fixed amount of neighbouring cells (*N*) in (5) and (6), linearly decreases the C_{din}. However, it also increases the number of required resources; the inverse is also true. By optimising the $\#ExpC$, the computation of the model is lower bounded by the topology independent C_{a+s}. Finally, we have to determine the time sharing factor, i.e. how many times a physical cell will be reused within a given time step (T_{step}). An upper bound for the time sharing factor can be calculated by taking into consideration the communication cycles (C_{com}) needed to send all dendrite and axon potential values through the tree network

$$\begin{aligned} TSF &< \begin{cases} \frac{C_{step} - C_{com}}{C_{phyc}} & C_{com} < C_{phyc} \\ \frac{C_{step} - C_{phyc}}{C_{com}} & \text{otherwise} \end{cases}, \\ C_{step} &= \frac{T_{step}}{Clk_{period}}. \end{aligned} \tag{7}$$

IV. Experimental setup and results

To verify the Inferior Olive architecture, multiple designs were implemented for multiple FPGA targets[6], based on (1)-(7). Firstly, the effects of the grouping factor are examined. The amount of physical cells is varied, each having its own exponent coprocessor. Grouping factors larger than four would result in poorer timing results, compared to smaller physical cell clusters that where placed in a larger tree network. However, as each physical cell cluster shares a single memory core, the amount of used BRAMs is reduced as well and, consequently, larger clusters could be routed more easily. By sharing each exponent coprocessor over 2 physical cells, resources were spared without having a large impact on the final timing delay. After the design is configured with the desired accuracy (32/64-bit), it is synthesised through the Vivado HLS tool to generate VHDL code, and test bench files[7]. To evaluate the proposed design, the synthesisable VHDL code is compiled with Modelsim, and the simulated axon voltages are then compared to the reference C model [7]. The optimal implementable hardware design timing to meet the 50µs 'real-time' constraint [14] for a single physical cell cluster are shown in Fig. 5. Given that the amount of computational cycles is fixed within a physical cell cluster for a given topology, the hardware designs closest to real-time timing constraint are scaled up by increasing the number of physical cell clusters in the tree network (Fig. 6). However, without routing tables in the tree network, all resulting potentials are sent in an all-2-all type fashion. The limit to the number of neurons that can give an output with an unbounded tree network can be calculated based on the number of output values that can be streamed within a certain time step. In the current design each output is given every 2 clock cycles (Δ). The theoretical maximum amount of outputs is found as

$$\#Ouput_{max} = \frac{C_{step} - C_{phyc}}{\Delta} - \Omega(\#Clusters) \tag{8}$$

where Ω is the amount of cycles it takes to deliver the first cluster output through the tree network to the I/O of the FPGA. By optimising the tree network with routing tables similar to those within the cluster control of the physical cell cluster, less cycles are used for communication, resulting in a stable time for increasing simulated cells. Finally, by combining the results from Fig. 6 with certain hardware limitations of the Virtex's and Spartan FPGAs,

[4]Within the dendrite calculation, Dend_{Comb} combines the 2 results after τ cycles.

[5]If an increase of the amount of Dend_{Net} operations were to coincide with the increase of exponent coprocessors, (5) would give the upper limit.

[6]The used FPGA targets are, the Virtex7: xc7vx550tffg1927-2 [16], the Virtex6: xc6vlx240tff1156-3 [17] and the Spartan6: xc6slx150tfgg676-3 [18].

[7]The test bench starts by initialising each cluster with a certain ID, each simulated cell with 19 neuron parameters followed by the initial state of each dendrite. Then after the topology of the NN is configured, the test bench pulses the start signal at T_{Step} time intervals.

TABLE I: Implementable cells on single FPGA with critical resources underlined.

Implementation						Resources (Absolute)				Resources (Total Utilisation)				Results	
FPGA	Accuracy	Clusters	PhyC	TSF	ExpC	LUT	FF	DSP	BRAM	LUT	FF	DSP	BRAM	SimC	Cost/SimC[b]
[10][a]	64b	NA	8	6	NA	240217	209153	1384	42	69.35%	30.19%	48.06%	1.78%	48	$ 144.00
Virtex7	64b	9	2	23	1	324670	202277	1215	233	93.72%	29.19%	42.18%	19.74%	414	$ 15.48
Virtex6	64b	1	7	20	2	124882	77712	634	122	82.86%	25.78%	82.55%	14.66%	140	$ 20.09
Spartan6	64b	1	1	8	1	23507	21297	113	27	25.51%	11.56%	62.78%	10.07%	8	$ 29.55
[7]	32b	NA	8	12	NA	251485	162217	1600	804	83%	27%	57%	78%	96	$ 51.22
Virtex7	32b	18	2	33	1	311808	190797	1008	557	90.01%	27.53%	35.00%	23.60%	1188	$ 5.39
Virtex6	32b	4	4	29	2	128670	85754	480	192	85.37%	28.48%	62.50%	23.08%	464	$ 6.06
Spartan6	32b	1	4	18	2	36772	23620	152	33	39.90%	12.82%	84.44%	12.31%	72	$ 3.28

[a] Only estimates are given in the previous design, built on the same Virtex7 FPGA as the current design.
[b] Reference costs were taken from [19].

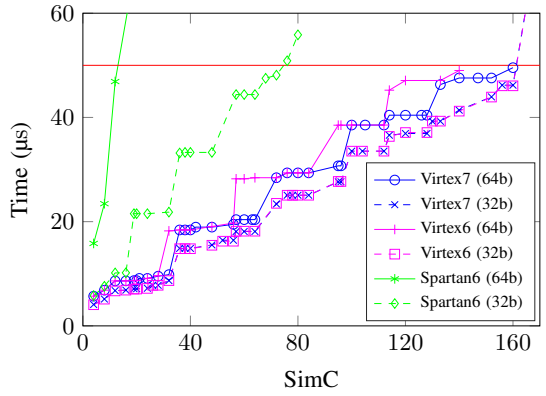

Fig. 5: Minimum calculation latency for several single-cluster implementable designs. The time step is represented by the straight line at 50µs.

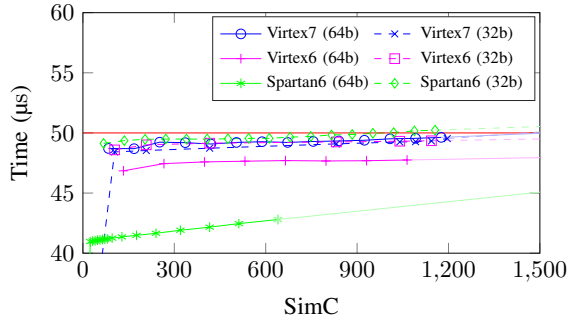

Fig. 6: Latency increase (communication) for several multiple-cluster designs based on an optimal single cluster design with simple routing through the tree network. The time step is represented by the straight line at 50µs.

Table I compares the results of the proposed design to previous designs [7], [10] for double and single floating-point precision. Each implementation has resulted in a more optimal use of critical resources, and a larger amount of simulated cells per FPGA. While, for 64-bit calculations, a modern FPGA seems desirable, Spartan6 has shown to be capable of cost-effectively running the Inferior Olive neuron calculations in 32-bit.

V. Conclusion

In this paper, we presented a 'real-time' hardware implementable set of inferior olive neurons. By modelling the design around a single coprocessor, the total number of cycles that each physical cell needs to simulate is bounded independently of the neuron network topology. Furthermore, by employing an open source floating-point exponent IP block, the design architecture can be used on more cost-effective FPGA targets. Finally, by applying the set of equations (1)-(7), future designs can be quickly optimised.

References

[1] J. P. Welsh and R. Llinas, "Some organizing principles for the control of movement based on olivocerebellar physiology," *Progress in Brain Research*, vol. 114, pp. 449–461, 1997.

[2] E. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[3] H. Shayani, P. J. Bentley, and A. M. Tyrrell, "Hardware implementation of a bio-plausible neuron model for evolution and growth of spiking neural networks on fpga," in *Adaptive Hardware and Systems, 2008. AHS '08. NASA/ESA Conference on*, June 2008, pp. 236–243.

[4] K. Cheung, S. R. Schultz, and W. Luk, "A large-scale spiking neural network accelerator for fpga systems," in *Artificial Neural Networks and Machine Learning–ICANN 2012*. Springer, 2012, pp. 113–120.

[5] Y. Zhang, E. R. J.P McGeehan, S. Kelly, and J. Nunez-Yanez, "Biophysically accurate floating point neuroprocessors for reconfigurable logic," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 599–608, 2013.

[6] M. Beuler, A. Tchaptchet, W. Bonath *et al.*, "Real-time simulations of synchronization in a conductance-based neuronal network with a digital fpga hardware-core," in *Artificial Neural Networks and Machine Learning–ICANN 2012*. Springer, 2012, pp. 97–104.

[7] G. Smaragdos, S. Isaza, M. F. van Eijk *et al.*, "Fpga-based biophysically-meaningful modeling of olivocerebellar neurons," in *IEEE International Symposium on Field-programmable Gate Arrays*, 2014, pp. 89–98.

[8] H. Nguyen, Z. Al-Ars, G. Smaragdos, and C. Strydis, "Accelerating complex brain-model simulations on gpu platforms," in *IEEE DATE*, 2015, pp. 974–979.

[9] D. Rodopoulos, G. Chatzikonstantis, A. Pantelopoulos *et al.*, "Optimal mapping of inferior olive neuron simulations on the single-chip cloud computer," in *IEEE SAMOS*, 2014, pp. 367–374.

[10] M. van Eijk, C. Galuzzi, A. Zjajo *et al.*, "Esl design of customizable real-time neuron networks," in *IEEE Biomedical Circuits and Systems Conference*, 2014, pp. 671–674.

[11] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University press, 2002.

[12] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[13] G. Kahn, "The semantics of a simple language for parallel programming," in *Information processing*, J. L. Rosenfeld, Ed. Stockholm, Sweden: North Holland, Amsterdam, 1974, pp. 471–475.

[14] P. Bazzigaluppi, J. R. De Gruijl, R. S. Van Der Giessen *et al.*, "Olivary subthreshold oscillations and burst activity revisited," *Frontiers in Neural Circuits*, vol. 6, pp. 1–13, 2012.

[15] F. de Dinechin and B. Pasca, "Floating-point exponential functions for DSP-enabled FPGAs," in *Field Programmable Technologies*, 2010, pp. 110–117.

[16] Xilinx, "DS183: Virtex-7 T and XT FPGAs Data Sheet," 2015.

[17] Xilinx, "DS152: Virtex-6 FPGA Data Sheet," 2015.

[18] Xilinx, "DS162: Spartan-6 FPGA Data Sheet," 2015.

[19] AVNET. Avnet express. Accessed on 19 September 2015. [Online]. Available: http://avnetexpress.avnet.com